

# Using Artificial Neural Networks to Construct a Meta-Model for the Evolution of Gait Patterns<sup>\*</sup>

Ingo Dahm<sup>1</sup> and Jens Ziegler<sup>2</sup>

<sup>1</sup> University of Dortmund, Computer Engineering Institute,  
Otto-Hahn-Strasse 4, 44227 Dortmund, Germany,  
ingo.dahm@uni-dortmund.de

<sup>2</sup> University of Dortmund, Department of Computer Science,  
Joseph-von-Fraunhofer-Strasse 20, 44227 Dortmund, Germany,  
jens.ziegler@uni-dortmund.de

## ABSTRACT

In this paper we suggest a novel method to approximate the fitness function of a genetic programming approach in order to develop fast and stable gait patterns for a quadruped robot. Therefore, gait patterns are classified by so called Signal Space Detectors. We show how a Signal Space Detector can extract information about the reliability of a classification. Finally, we demonstrate how this information can be used to replace conventional time-consuming fitness modules like real-world tournaments or offline simulations.

## 1 MOTIVATION

Genetic programming [1] is expected to be a capable solution for designing fast and robust walking patterns for legged robots [2–5]. In fact, using this approach we have evolved very fast and stable patterns in the past [6]. The genetic programming approach is easy to implement since the fitness function here is given implicitly by the speed of the robots using a specific gait pattern. The best performing individual can be found easily by executing foottraces. Unfortunately, such tournaments suffer from heavy wearout during the different stages of evolution caused by many evaluations. Therefore, different techniques are used to reduce the number of tournaments. An efficient method is to use physical simulation as a fitness module. In the past, we have made promising experiences using a physical simulation of a biped robot [7]. Nevertheless, the implementation of such simulations can be very complex and expensive. Furthermore, in some cases, the characteristics of motors, joints or gears might be unknown. Therefore, a physical simulation is not implementable in general. Instead, we suggest the use of a meta-model that can be generated automatically by an artificial neural network.

## 2 ARTIFICIAL NEURAL NETWORKS AS FITNESS MODULE

We used ERS 2100 quadruped robots for our studies. That robots are manufactured by SONY and are widely-used for robot soccer purposes [8]. They provide three degrees of freedom (DOF) per leg. The movements are controlled by a 27-parameter walking engine [9]. An overview over the walking engine parameters is given in Table 1. Obviously, an 27-dimensional vector can be interpreted by the walking engine as a control program for the robot.

---

<sup>\*</sup> This work was supported by the Deutsche Forschungsgemeinschaft (DFG)

**Table 1. Parameter set for the “Inverse Kinematic Walking Engine” [9]**

Number	Cumulated	Parameter	Type	Description
1	1	footmode	binary	circular or tetragonal path of foot movement
3	4	foreHeight, foreWidth, foreCenter	float	foreleg position when standing
3	7	hindHeight, hindWidth, hindCenter	float	hind leg position when standing
2	9	foreFootTilt, hindFootTilt	float	tilt angles of foot movement paths to y-axis
2	11	foreFootLift, hindFootLift	float	maximal distance between foot and ground
3	14	legSpeedFactorX legSpeedFactorY legSpeedFactorR	float	ratio of step sizes of fore legs and hind legs
2	16	maxStepSizeX, maxStepSizeY	float	step length
3	19	maxSpeedXChange maxSpeedYChange maxRotationChange	float	frequency of motion request updates
1	20	counterRotation	float	angle to x-axis of superpositioned rotation
1	21	stepLen	int	quantization step size of the movement
1	22	groundPhase	float	ratio of foot lifting time and standing
1	23	liftPhase	float	time to entire time for a single step
4	27	headTilt, headPan, headRoll, mouth	float	head position (constant) for the walk

## 2.1 The Signal Space Concept

For our evolutionary approach, we use individuals that consist of  $M = 27$  variables. Therefore, the individuals are characterized by  $M$ -dimensional vectors  $\mathbf{V}_i = (v_1, v_2, \dots, v_M)$ . That vectors can be interpreted geometrically as coordinates of points in an  $M$ -dimensional space. Since each vector represents a walking pattern, every  $\mathbf{V}_i$  implicates a walking speed. Depending on this specific speed, fast and slow individuals can be separated into different classes  $C^{sup}$  and  $C^{inf}$ , whereby superior individuals (e.g. faster ones) are in  $C^{sup}$ . Inferior individuals are classified to be member of  $C^{inf}$ . All  $\mathbf{V}_i \in C^{inf}$  are discriminated against superior individuals.

A single individual can be interpreted as a point in an  $M$ -dimensional space, thus similar individuals form clusters in that signal space. Therefore, the classification can be done by partitioning the signal space. If a subspace  $S_j$  contains all  $\mathbf{V}_i \in C^{inf}$ , then the remaining space contains all  $\mathbf{V}_i \in C^{sup}$ . Geometrically spoken, the space becomes divided into regions such that all superior individuals (e.g. fast gait patterns) are located in different regions than the inferior individuals.

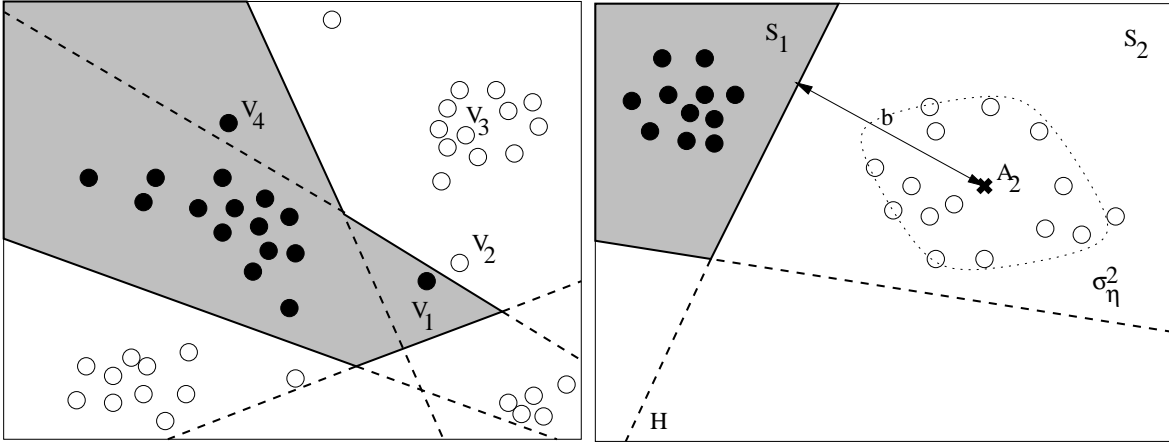
If clusters can be approximated by a center point and superpositioned additive white Gaussian distributed noise (AWGN), then the boundaries are given implicitly since the subspaces are given by Voronoi Regions [10]. For example, in the two-dimensional case the clusters must form circular shapes. For  $M = 3$ , the clusters must be approximatable by spheres.

In all other cases, the boundaries have more complex shapes which can be approximated by sets of several hyperplanes. Such a situation is illustrated in Figure 1 exemplarily: the superior individuals (black) and inferior individual are divided by hyperplanes (bold lines). The region which indicates, that individuals are superior ( $\mathbf{V}_i \in C^{sup}$ ) is shaded gray.

## 2.2 Implementing the Signal Space Detector as Artificial Neural Network

The vector  $V_i$  can be uniquely classified by associating to it a subspace  $S_j$  as shown in section 2.1. This can be done by calculating the distances to all bounding hyperplanes of the regions that characterize  $C^{sup}$  and  $C^{inf}$  respectively. Since the distance between point and hyperplane is given by the dot-product of the plane’s normal vector<sup>c</sup> and the point itself, such an implementation is of low complexity. In recent publications, that efficient implementation is known as

<sup>c</sup> The normal vector must be normalized to 1.0



**Fig. 1. A set of individuals in a two-dimensional space. The subspace of superior individuals is shaded gray (left). Estimating the reliability information referring to hyperplane  $H$  using the notation as given in section 2.**

Signal Space Detector (SSD) [11, 12]. A hyperplane in the SSD can be estimated by a conventional perceptron [6, 13]. Such a perceptron multiplies its weight vector  $\mathbf{w} = (w_1, w_2, \dots, w_M)^T$  with the input vector and adds a bias  $w_{M+1}$ . If  $\mathbf{V}_i$  is used as input vector, then  $\mathbf{w}$  can be interpreted as the normal vector of a bounding hyperplane  $H$ .

$$H : \begin{pmatrix} w_1 \\ w_2 \\ \dots \\ w_M \end{pmatrix} \cdot \mathbf{V}_i + w_{M+1} = 0 \quad (1)$$

After computing the dot-product of both vectors, a perceptron weights the result by an activation function  $A$ . Thus, the perceptrons output  $p$  is given by Equation (2).

$$p = A \left( \begin{pmatrix} w_1 \\ w_2 \\ \dots \\ w_{M+1} \end{pmatrix} \cdot (\mathbf{V}_i, 1)^T \right) \quad (2)$$

For simplified learning rules, typically a sigmoid activation function as illustrated in Eq. (3) is used [14]. If necessary, the distance between  $\mathbf{V}_i$  and  $H$  represented by the perceptron can be computed by normalizing the weight vector to  $|\mathbf{w}| = 1$  and applying the inverse function  $A^{-1}$  after Eq. (4).

$$A : x \rightarrow \frac{1}{1 + e^{c(t-x)}} \quad (3)$$

$$A^{-1} : x \rightarrow t - \frac{1}{c} \cdot \ln \left( \frac{1}{x} - 1 \right) \quad (4)$$

Obviously, it is not difficult to classify individuals using a signal space approach. The decision planes can be found easily by using a multi-layer perceptron approach and a back-propagation learning algorithm. Unfortunately, that approach suffers from low classification accuracy of the artificial neural network. We observed that more than 20 percent of tournaments are classified incorrectly (see Fig. 2).

### 2.3 Improving Classification Accuracy by Confidence-Estimation

As illustrated in Figure 1, some individuals (like  $\mathbf{V}_1$  and  $\mathbf{V}_2$ ) are located closely to a decisive plane. Other individuals (like  $\mathbf{V}_3$ ) are located near the center of a cluster of individuals and have a long distance to the bounding hyperplanes. Small changes of the normal vectors of the decision planes affect especially the classification of those individuals which are located near the hyperplanes. Actually, the normal vector varies in a small range after each step in the training phase. Therefore, it can be justified to assign a high reliability to the classification of  $\mathbf{V}_3$  and a low classification accuracy to  $\mathbf{V}_1$  and  $\mathbf{V}_2$ .

In [15], the proof of this approach is given in detail. Furthermore, a formula to calculate the confidence of a decision using a special signal space detector ( $S^3D$ ) can be found in this paper. For this purpose, a so called ‘‘admissible signal point’’  $\mathbf{A}_j$  for each subspace  $S_j$  must be found. Afterwards, all individuals in the same subspace can be modeled by adding colored noise  $\eta$  with a variance  $\sigma_\eta^2$  to  $\mathbf{A}_j$  as illustrated by Equation (5) and the right part of Figure 1.

$$\mathbf{V}_i = \mathbf{A}_j + \eta \quad \forall i \text{ with } (\mathbf{V}_i, \mathbf{A}_j) \in S_j \quad (5)$$

Then, the reliability  $L$  of the decision is given by the normalized dot-product of  $\mathbf{V}_i$  and the hyperplanes normal vector  $\mathbf{w}$  after Eq. (6). Note, that  $b$  indicates the distance from  $\mathbf{A}_j$  to the decisive hyperplane  $H$ .

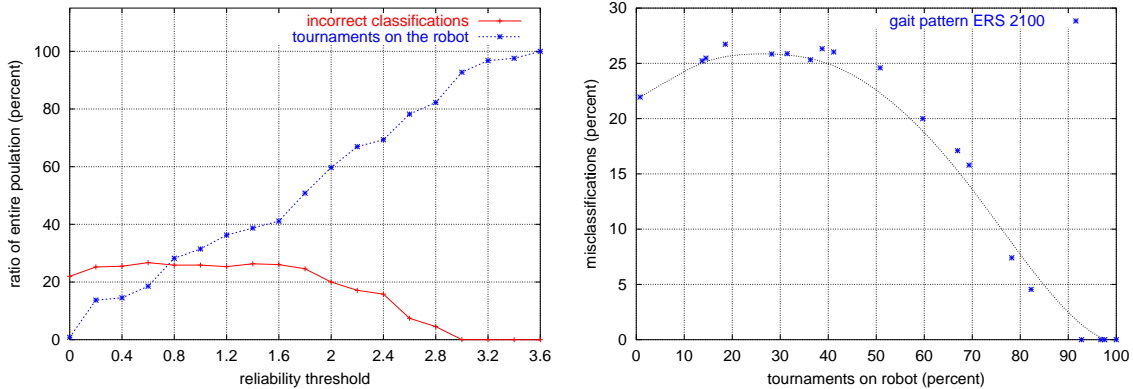
$$L = \frac{b}{|\mathbf{w}| \cdot \sigma_\eta^2} \cdot (\mathbf{w} \cdot \mathbf{V}_i) = \frac{b}{|\mathbf{w}| \cdot \sigma_\eta^2} \cdot A^{-1}(p) \quad (6)$$

Thus, the knowledge of  $\mathbf{A}_j$  is essential to calculate the confidence of the classification. If there is a simulation model of the robot, such admissible signal points can be extracted out of that model. Else, the position of the admissible point must be estimated. For our studies we use the following basic approach: After training the artificial neural network, we compute the focal point of all points in the same subspaces  $S_j$ . From this focal point  $\mathbf{A}_j$ , we calculate the distance to all other observed points in the same subspace in order to estimate the variance  $\sigma_\eta^2$ . Furthermore, we calculate the distance  $b$  from  $\mathbf{A}_j$  to each adjacent hyperplane  $H$ . This can be done efficiently in the first layer of a multi-layer perceptron (MLP) network. An example is illustrated in Figure 1. As presented in [15], some hyperplanes do not contribute to the decision. In Figure 1, this situation can be observed for  $\mathbf{V}_4$ : The dotted hyperplane is close to the point. This would implicate a low reliability. On the other hand,  $\mathbf{V}_4$  has a long distance to the white shaded area which indicates a high confidence. Therefore, we must establish a filtering technique to discriminate between decisive and non-decisive hyperplanes. In the  $S^3D$ -architecture, decisive hyperplanes are found out by inverting the result of each plane and propagate it through a boolean algebra. Here, we have no boolean logic. So we use another two layers in the MLP.

There, we invert the result of the previous perceptrons weight-function and propagate it through the neural network. If the final result changes, the hyperplane is claimed to be decisive. Decisive hyperplanes contribute to the overall confidence calculation. As there can be a couple of decisive hyperplanes, we have to combine all their reliabilities to an overall confidence. This is done by the min-rule: We determine the confidence for each decisive perceptron. The overall confidence is given by the minimum of all calculated reliabilities.

### 3 RESULTS

The suggested technique takes to two major advantages. First, it reduces the robot wearout during training phase. Second, the learning process can be accelerated significantly, since real-world tournaments can be skipped. Thus, the population size can be increased, while wearout and duration of the evolution decrease simultaneously. This intends to further enhance the quality of walking patterns.



**Fig. 2. Reliability threshold vs. classification correctness and real-world tournaments (left). Number of misclassifications vs. real-world tournaments (right). Results achieved by evolving gait patterns using “Inverse Kinematic Walking Engine [9]”**

#### 3.1 Replacing Real-World Tournaments by Artificial Neural Networks

By increasing the reliability threshold, which indicates if an individuals fitness is estimated correctly by the MLP-approach, the ratio of misclassifications can be reduced significantly. Figure 2 shows how many tournaments must be performed on the robots (y-axis). This is indicated by the number classifications which show a reliability less than the given threshold (x-axis). Additionally, the number of incorrect classifications is represented, too. For example, with a reliability threshold of  $L = 2.6$  about 21.8% of the tournaments can be saved. Furthermore, only 7.4% of the classifications are incorrect.

Without that offline-fitness estimation, we evolved a walk, which is 28% faster than a reference (hand-optimized) gait pattern by performing 285 tournaments on the ERS 2100 robot.

With the suggested methodology, we approximated the results of the foottraces by the enhanced artificial neural network. 10% of the unreliable decisions have been selected randomly to be performed on the robots. Thereby, we evolved a gait pattern which is about 20% faster than the reference walk. The wearout was reduced significantly, since only 34 tournaments had to be evaluated.

### 4 CONCLUSION

We observed that the quality of the evolved gaits depends on internal network parameters and the meta-model complexity. That correlation is addressed in the paper for SONY ERS 2100 robots and further investigations will show how that approach performs under different hardware conditions.

We have to analyze if and how the suggested meta-model can substitute real-world tournaments completely. This aim seems realistic, since up to 99.9% of the tournament results can be predicted correctly. To achieve this, the population size is reduced significantly as all unreliable classifications are not used for the evolutionary algorithm. Thus, we have to investigate in general if discriminating low reliable patterns against unreliable decisions prevents from finding optimal gaits. To do this, we have to examine how neural networks and signal space parameters can automatically be adjusted to the actual robot control program.

## 5 ACKNOWLEDGMENTS

We thank the German United Team, especially the “Ruhrpott Hellhounds” for their excellent software support. Further, we thank Stefan Schmerbeck for his valuable comments about confidence estimation. Third, we thank Max Risler and the “Darmstadt Dribbling Dackels” for their advantageous walking engine.

## REFERENCES

1. **J. R. Koza**, *Genetic Programming*. Cambridge, MA: MIT Press, 1992.
2. **J. Busch, J. Ziegler, C. Aue, A. Ross, D. Sawitzki, and W. Banzhaf**, “Automatic generation of control programs for walking robots using genetic programming,” in *Proceedings of the 5th European Conference on Genetic Programming* (J. A. Foster, E. Lutton, J. Miller, C. Ryan, and A. G. B. Tettamanzi, eds.), vol. 2278 of *Lecture Notes in Computer Science*, pp. 258–268, Springer, New York, 2002.
3. **G. F. Spencer**, “Automatic generation of programs for crawling and walking,” in *Advances in Genetic Programming* (K. E. Kinnear, Jr., ed.), ch. 15, pp. 335–353, MIT Press, 1994.
4. **M. A. Lewis, A. H. Fagg, and A. Solidum**, “Genetic programming approach to the construction of a neural network control of a walking robot,” in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, (Nice, France), pp. 2618–2623, Electronica Bks, 1992.
5. **G. S. Hornby, M. Fujita, S. Takamura, T. Yamamoto, and O. Hanagata**, “Autonomous evolution of gaits with the sony quadruped robot,” in *Proceedings of the Genetic and Evolutionary Computation Conference* (W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, eds.), vol. 2, (Orlando, Florida, USA), pp. 1297–1304, Morgan Kaufmann, 13-17 1999.
6. **I. Dahm and J. Ziegler**, “Adaptive methods to improve self-localization in robot soccer,” in *RoboCup Symposium, Fukuoka*, 2002.
7. **J. Ziegler, J. Barnholt, J. Busch, and W. Banzhaf**, “Automatic evolution of control programs for a small humanoid walking robot,” in *International Conference on Climbing and Walking CLAWAR*, 2002.
8. **M. Veloso, W. Uther, M. Fujita, M. Asada, and H. Kitano**, “Playing soccer with legged robots,” in *International Conference on Intelligent Robots and Systems IEEE/RSJ*, pp. 437–442, October 1998.
9. **M. Risler**, “InvKinWalkingEngine - kurze Beschreibung der Parameter,” Technical Note, Technische Universität Darmstadt, 2002.
10. **T. Jeon and J. Moon**, “A Systematic Approach to Signal Space Detection,” *IEEE Trans. Magn.*, vol. 33, pp. 2737–2739, September 1997.
11. **B. Brickner and J. Moon**, “A High Dimensional Signal Space Implementation of FDTs/DF,” *IEEE Transactions on Magnetics*, vol. 32, pp. 3941–3943, September 1996.

12. **Y. Kim and J. Moon**, “Low Complexity Signal Space Detector for (1,7)-Coded Partial Response Channels,” *IEEE Trans. Magn.*, vol. 34, pp. 1928–1930, July 1998.
13. **J. A. Anderson**, *An Introduction to Neural Networks*. No. ISBN 0-262-01144-1, MIT Press. Boston, 1995.
14. **S. Haykin**, *Neural Networks: A Comprehensive Foundation*. No. ISBN 0-02-352761-7, Macmillan College Publishing Company Inc., 1994.
15. **S. Schmerbeck and G. Stromberg**, “Soft-output signal space detectors (s3d),” Tech. Rep. 0102, Computer Engineering Institute (CEI), University of Dortmund, 2002.

© 2002 by University of Dortmund